Symposium On Fractional Signals and Systems

IPC, Coimbra, Portugal
4 - 5 November 2011

Fractional Order Darwinian Particle Swarm Optimization

Micael S. Couceiro[1], N. M. Fonseca Ferreira[1], J. A. Tenreiro Machado[2]

[1] RoboCorp, Department of Electrotechnics Engineering,
Engineering Institute of Coimbra, Portugal
{micael,nunomig}@isec.pt

[2] Department of Electrotechnics Engineering,
Engineering Institute of Porto, Portugal
jtm@isep.ipp.pt

Abstract *— The Darwinian Particle Swarm Optimization (DPSO) is an evolutionary algorithm that extends the Particle Swarm Optimization (PSO) using natural selection, or survival-of-the-fittest, to enhance the ability to escape from local optima. This paper presents a method for controlling the convergence rate of the DPSO using fractional calculus (FC) concepts. The fractional order (FO) DPSO*, denoted as *FO-DPSO*, *is tested using several well-known functions and the relationship between the fractional order velocity and the convergence of the algorithm is observed.*

Keywords *— fractional calculus, DPSO, evolutionary algorithm.*

1 Introduction

Bio-inspired algorithms have been employed in situations where conventional optimization techniques cannot find a satisfactory solution, for example when the function to be optimized is discontinuous, non-differentiable, and/or presents too many nonlinearly related parameters [1]. One of the most well-known bio-inspired algorithms used in optimization problems is the Particle Swarm Optimization (*PSO*) which basically consists on a machine-learning technique loosely inspired by birds flocking in search of food [2]. More specifically, it consists of a number of particles that collectively move on the search space in search of the global optimum. The Darwinian Particle Swarm Optimization (*DPSO*), an evolutionary algorithm that extends the *PSO* using natural selection, was developed to enhance the ability of the *PSO* to escape from local optima [3].

The theory of Fractional Calculus (*FC*) is a useful mathematical tool for applied sciences [4]. In fact, *FC* has played a very important role increasing the performance of several algorithms used in modelling, curve fitting, filtering, pattern recognition, edge detection, identification, stability, controllability, observability and robustness.

Therefore, this paper proposes a fractional-order (*FO*) *DPSO* using fractional calculus to control the convergence rate of the algorithm. Section two presents the state-of-the-art of the several Particle Swarm Optimization (*PSO*) main variants mainly focusing on the Darwinian Particle Swarm Optimization (*DPSO*) developed by [3]. Section three generalizes the *DPSO* to a fractional order. Experimental results for the *FO-DPSO* are presented in section four. Finally, in section five outlines the main conclusions.

2 A Survey on the Particle Swarm Optimization Techniques

The original *PSO* was developed by Eberhart and Kennedy in 1995 [2] and it is based on social and computer science. The *PSO* basically takes advantages on the swarm intelligence concept, which is the property of a system, whereby the collective behaviors of unsophisticated agents that are interacting locally with their environment, create coherent global functional patterns [5]. Imagine a flock of birds where each bird cries at an intensity proportional to the amount of food that it finds at its current location. At the same time each bird can perceive the position of neighboring birds and can tell which of the neighboring birds emits the loudest cry. There is a good chance that the flock will find a spot with the highest concentration of food if each bird follows a trajectory that combines three directions: *i*) keep flying in the same direction; *ii*) return to the location where it found the highest concentration of insects so far; and *iii*) move toward the neighboring bird that cries the loudest [1]. In the traditional *PSO*, the candidate solutions are called particles. These particles travel through the search space to find an optimal solution, by interacting and sharing information with neighbor particles, namely their individual best solution (local best) and computing the neighborhood best. Also, in each step of the procedure, the global best solution obtained in the entire swarm is updated. Using all of this information, particles realize the locations of the search space where success was obtained, and are guided by these successes. In each step of the algorithm (Algorithm 1), a fitness function is used to evaluate the particle success. To model the swarm, each particle n moves in a multidimensional space according to position (x_t^n) and velocity (v_t^n) values which are highly dependent on local best ($\breve{x}_t^n$), neighborhood best ($\breve{n}_t^n$) and global best ($\breve{g}_t^n$) information:

$$v_{t+1}^n = wv_t^n + \rho_1 r_1(\breve{g}_t^n - x_t^n) + \rho_2 r_2(\breve{x}_t^n - x_t^n) + \rho_3 r_3(\breve{n}_t^n - x_t^n) \quad (1)$$

$$x_{t+1}^n = x_t^n + v_{t+1}^n \quad (2)$$

The coefficients w, ρ_1, ρ_2 and ρ_3 assign weights to the inertial influence, the global best, the local best and the neighborhood best when determining the new velocity, respectively. Typically, the inertial influence is set to a value slightly less than 1. ρ_1, ρ_2 and ρ_3 are constant integer values, which represent "cognitive" and "social" components. However, different results can be obtained by assigning different influences for each component. For example, several works do not consider the neighborhood best and ρ_3 is set to zero. Depending on the application and the characteristics of the problem, tuning these parameters properly will lead to better results. The parameters r_1, r_2 and r_3 are random vectors with each component generally a uniform random number between 0 and 1. The intent is to multiply a new random component per velocity dimension, rather than multiplying the same component with each particle's velocity dimension.

In the beginning, the particles' velocities are set to zero and their position is randomly set within the boundaries of the search space (Algorithm 1). The local, neighborhood and global bests are initialized with the worst possible values, taking into account the nature of the problem. There are other few parameters that need to be adjusted: *i*) population size – very important to optimize to get overall good solutions in acceptable time; and *ii*) stopping criteria – it can be a predefined number of iterations without getting better results or other criteria, depending on the problem.

Initialize swarm (Initialize x_t^n, v_t^n, $\breve{x}_t^n$, $\breve{n}_t^n$ and $\breve{g}_t^n$)
Loop:
    for all particles
        Evaluate the fitness of each particle
        Update $\breve{x}_t^n$, $\breve{n}_t^n$ and $\breve{g}_t^n$
        Update v_{t+1}^n and x_{t+1}^n
    end
until stopping criteria (convergence)

Algorithm 1: Traditional *PSO* Algorithm

PSO reveals an effect of implicit communication between particles (similar to broadcasting) by updating neighborhood and global information, which affects the velocity and consequent position of particles. Also, there is a stochastic exploration effect due to the introduction of the random multipliers (r_1, r_2 and r_3). The *PSO* has been successfully used in many applications such as robotics [6-9], electric systems [10] and sport sciences [11].

However, a general problem with the *PSO* and other optimization algorithms is that of becoming trapped in a local optimum such that it may work well on one problem but may fail on another problem. In order to overcome this problem many authors have suggested other adjustments to the parameters of the *PSO* algorithm combining Fuzzy logic (*FAPSO*) where the inertia weight w is dynamically adjusted using fuzzy "*IF-THEN*" [12] rules or Gaussian approaches (*GPSO*) where the inertia constant w is no longer needed and the acceleration constants ρ_1, ρ_2 and ρ_3 are replaced by random numbers with Gaussian distributions [13].

More recently, Pires *et al.* used fractional calculus to control the convergence rate of the *PSO* [14]. The authors rearrange the original velocity equation (1) in order to modify the order of the velocity derivative. This paper tries to control the convergence rate of an evolutionary version of the *PSO* based on Pires *et al.* work since the well-succeeded variants of the *PSO* are the ones based on evolutionary techniques [5].

Many authors have considered incorporating selection, mutation and crossover, as well as the differential evolution (*DE*), into the *PSO* algorithm. The main goal is to increase the diversity of the population by either preventing the particles to move too close to each other and collide [15-16] or to self-adapt parameters such as the constriction factor, acceleration constants [17], or inertia weight [18].

The fusion between Genetic Algorithms (*GA*) and the *PSO* originated the *GA-PSO* [19] which combines the advantages of swarm intelligence and a natural selection mechanism, such as *GA*, in order to increase the number of highly evaluated agents, while decreasing the number of lowly evaluated agents at each iteration step. Similar to this last one, the *EPSO* is an evolutionary approach that incorporates a selection procedure to the original *PSO* algorithm, as well as self-adapting properties for its parameters. This algorithm adds a tournament selection method used in evolutionary programming (*EP*) [20]. Based on the *EPSO*, a differential evolution operator has been proposed to improve the performance of the algorithm in two different ways. The first one [21] applies the differential evolution operator to the particle's best position to eliminate the particles falling into local minima (*DEPSO*) while the second one [22] applies it to find the optimal parameters (inertia and acceleration constants) for the canonical *PSO* (*C-PSO*).

In search of a better model of natural selection using the *PSO* algorithm, the Darwinian Particle Swarm Optimization (*DPSO*) was formulated by [3], in which many swarms of

test solutions may exist at any time. Each swarm individually performs just like an ordinary *PSO* algorithm with some rules governing the collection of swarms that are designed to simulate natural selection. Despite the similarities between the *PSO* and Genetic Algorithms (*GA*s) like randomly generated population, fitness function evaluation, population update, search for optimality with random techniques and not guaranteeing success; *PSO* does not use genetic operators like crossover and mutation thus, not being considered an evolutionary technique. On the other hand, the Darwinian Particle Swarm Optimization (*DPSO*) extends the *PSO* to determine if natural selection (Darwinian principle of survival of the fittest) can enhance the ability of the *PSO* algorithm to escape from local optima. The idea is to run many simultaneous parallel *PSO* algorithms, each one a different swarm, on the same test problem and a simple selection mechanism is applied. When a search tends to a local optimum, the search in that area is simply discarded and another area is searched instead. In this approach, at each step, swarms that get better are rewarded (extend particle life or spawn a new descendent) and swarms which stagnate are punished (reduce swarm life or delete particles). To analyze the general state of each swarm, the fitness of all particles is evaluated and the neighborhood and individual best positions of each of the particles are updated. If a new global solution is found, a new particle is spawned. A particle is deleted if the swarm fails to find a fitter state in a defined number of steps (Algorithm 2).

Some simple rules are followed to delete a swarm, delete particles, and spawn a new swarm and a new particle: *i*) when the swarm population falls below a minimum bound, the swarm is deleted; and *ii*) the worst performing particle in the swarm is deleted when a maximum threshold number of steps (search counter $SC_C^{\max}$) without improving the fitness function is reached. After the deletion of the particle, instead of being set to zero, the counter is reseted to a value approaching the threshold number, according to:

$$SC_C(N_{kill}) = SC_C^{\max}\left\lfloor 1 - \frac{1}{N_{kill}+1} \right\rfloor \tag{3}$$

With N_{kill} being the number of particles deleted from the swarm over a period in which there was no improvement in fitness. To spawn a new swarm, a swarm must not have any particle ever deleted and the maximum number of swarms must not be exceeded. Still, the new swarm is only created with a probability of $p = f / NS$, with f a random number in [0,1] and *NS* the number of swarms. This factor avoids the creation of newer swarms when there are large numbers of swarms in existence. The parent swarm is unaffected and half of the parent's particles are selected at random for the child swarm and half of the particles of a random member of the swarm collection are also selected. If the swarm initial population number is not obtained, the rest of the particles are randomly initialized and added to the new swarm. A particle is spawned whenever a swarm achieves a new global best and the maximum defined population of a swarm has not been reached.

Main Program Loop	**Evolve Swarm Algorithm**
For each swarm in the collection Evolve the swarm (Evolve Swarm Algorithm: right) Allow the swarm to spawn Delete "failed" swarms	For each particle in the swarm Update Particles' Fitness Update Particles' Best Move Particle If swarm gets better Reward swarm: spawn particle: extend swarm life If swarm has not improved Punish swarm: possibly delete particle: reduce swarm life

Algorithm 2: *DPSO* Algorithm

Like the *PSO*, a few parameters also need to be adjusted to run the algorithm efficiently: *i*) initial swarm population; *ii*) maximum and minimum swarm population; *iii*) initial number of swarms; *iv*) maximum and minimum number of swarms; and *v*) stagnancy threshold. In estimation problems previously studied in [11] and robotic exploration strategies developed in [23] the *DPSO* has been successfully compared with the *PSO* showing a superior performance.

Later on, the results obtained using the proposed *FO-DPSO* will be compared with the *FO-PSO* developed by [14] and discussed. Next chapter presents the use of the *FC* to control the convergence rate of the *DPSO*.

3 Fractional-Order Darwinian Particle Swarm Optimization

In this section, a new method to control the *DPSO* algorithm based on Pires *et al.* approach to the traditional *PSO* [14] is introduced and denoted as *FO-DPSO*.

Fractional calculus (*FC*) has attracted the attention of several researchers [4;24-25], being applied in various scientific fields such as engineering, computational mathematics, fluid mechanics, among others [26-29]. The Grünwald–Letnikov definition based on the concept of fractional differential with $\alpha \in C$ of the signal $x(t)$, is given by:

$$D^{\alpha}[x(t)] = \lim_{h \to 0} \left[\frac{1}{h^{\alpha}} \sum_{k=0}^{+\infty} \frac{(-1)^k \Gamma(\alpha+1) x(t-kh)}{\Gamma(k+1)\Gamma(\alpha-k+1)} \right] \tag{4}$$

where Γ is the gamma function.

An important property revealed by the Grünwald–Letnikov equation (4) is that while an integer-order derivative just implies a finite series, the fractional-order derivative requires an infinite number of terms. Therefore, integer derivatives are 'local' operators while fractional derivatives have, implicitly, a 'memory' of all past events. However, the influence of past events decreases over time.

Based on equation (4), a discrete time implementations expression can be defined as:

$$D^{\alpha}[x(t)] = \frac{1}{T^{\alpha}} \sum_{k=0}^{r} \frac{(-1)^k \Gamma(\alpha+1) x(t-kT)}{\Gamma(k+1)\Gamma(\alpha-k+1)} \tag{5}$$

where T is the sampling period and r is the truncation order.

The characteristics revealed by fractional calculus make this mathematical tool well suited to describe phenomena such as irreversibility and chaos because of its inherent

memory property. In this line of thought, the dynamic phenomena of particle's trajectory configure a case where fractional calculus tools fit adequately.

Considering the inertial influence of equation (1) $w = 1$, assuming $T = 1$ and based on [14] work, the following expression can be defined:

$$D^{\alpha}[v_{t+1}^{n}] = \rho_1 r_1(\breve{g}_t^n - x_t^n) + \rho_2 r_2(\breve{x}_t^n - x_t^n) + \rho_3 r_3(\breve{n}_t^n - x_t^n) \tag{6}$$

Considering the first $r = 4$ terms of differential derivative given by (5), equation (6) can be rewritten as (7):

$$\begin{aligned} v_{t+1}^{n} = \alpha v_t^n + \frac{1}{2}\alpha v_{t-1}^n + \frac{1}{6}\alpha(1-\alpha)v_{t-2}^n + \frac{1}{24}\alpha(1-\alpha)(2-\alpha)v_{t-3}^n \\ + \rho_1 r_1(\breve{g}_t^n - x_t^n) + \rho_2 r_2(\breve{x}_t^n - x_t^n) + \rho_3 r_3(\breve{n}_t^n - x_t^n) \end{aligned} \tag{7}$$

The *FO-DPSO* will be evaluated in next section using equation (7) for all particles in all swarms. The *DPSO* is then considered as being a particular case of the *FO-DPSO* when $\alpha = 1$ (without 'memory').

4 Experimental Results

This section presents experimental results of the proposed *FO-DPSO*. Also, in order to compare this approach with Pires *et al.* approach [14] the same test functions and parameters are used as depicted in Table 1. Table 1 also shows the specific parameters of the *DPSO* algorithm.

	Min	*Initial*	*Max*
Number of Simulations	-	*201*	-
Number of Iterations	-	*200*	-
Coefficients $\rho_1 = \rho_2 = \rho_3$	-	*0.8*	-
Swarm Population	*3*	*4*	*5*
Number of Swarms	*1*	*2*	*3*
Stagnancy Threshold	-	*10*	-
Optimization Functions f_j [30]	*1- Bohachevsky 1* *2- Colville* *3- Drop wave* *4- Easom* 5- *Rastrigin*		

Table 1: Specifications of the algorithm and optimization functions.

The median of the fitness evolution of the best global particle is taken as the system output, for each value in the set $\alpha = \{0, 0.1, \ldots, 1\}$. In Figs. 1–5, the results can be seen for the adopted optimization functions f_j , $j = \{1, \ldots, 5\}$.

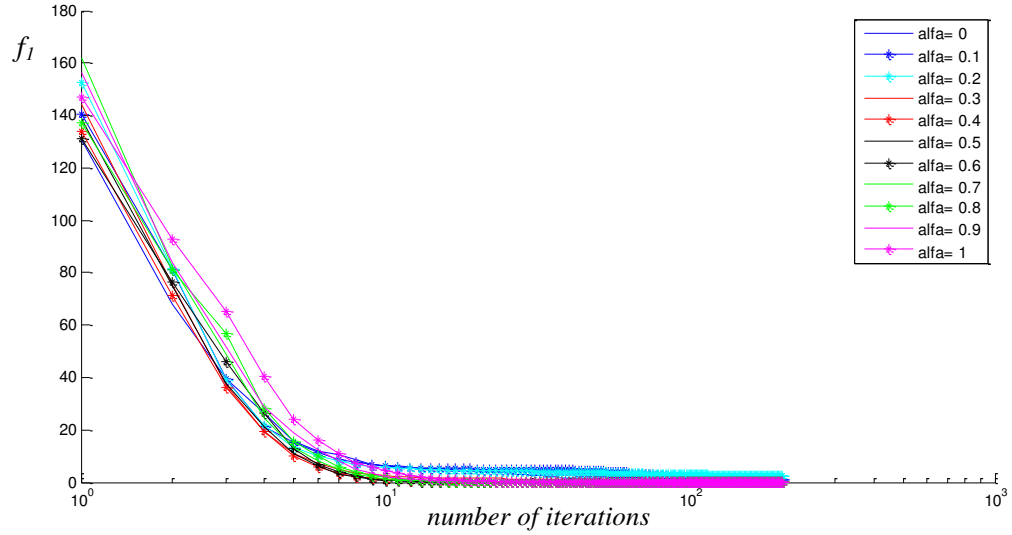

Figure 1: Evolution of the *Bohachevsky 1* function changing α.

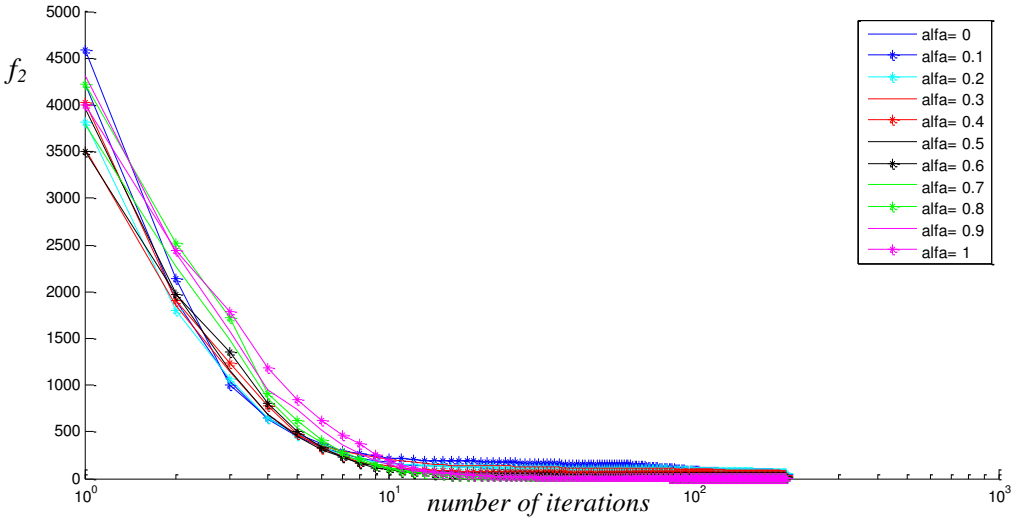

Figure 2: Evolution of the *Colville* function changing α.

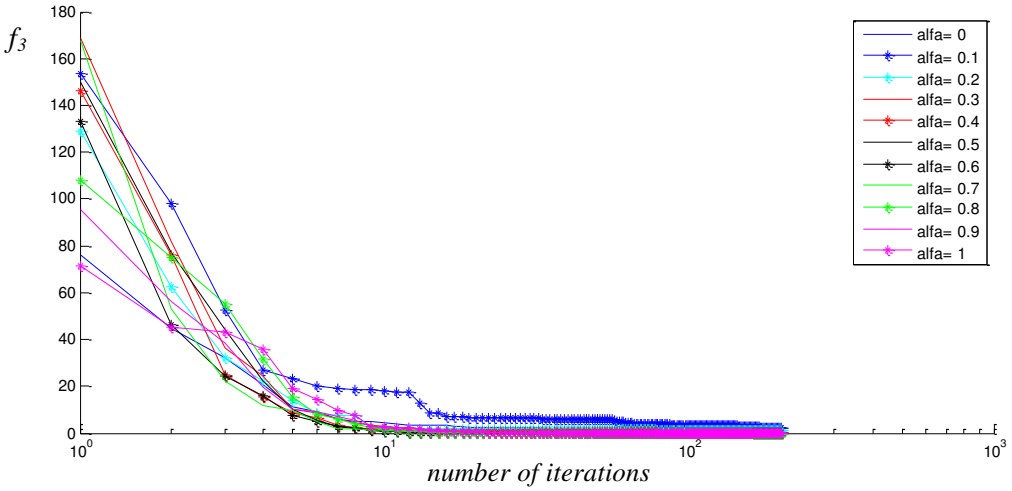

Figure 3: Evolution of the *Drop wave* function changing α.

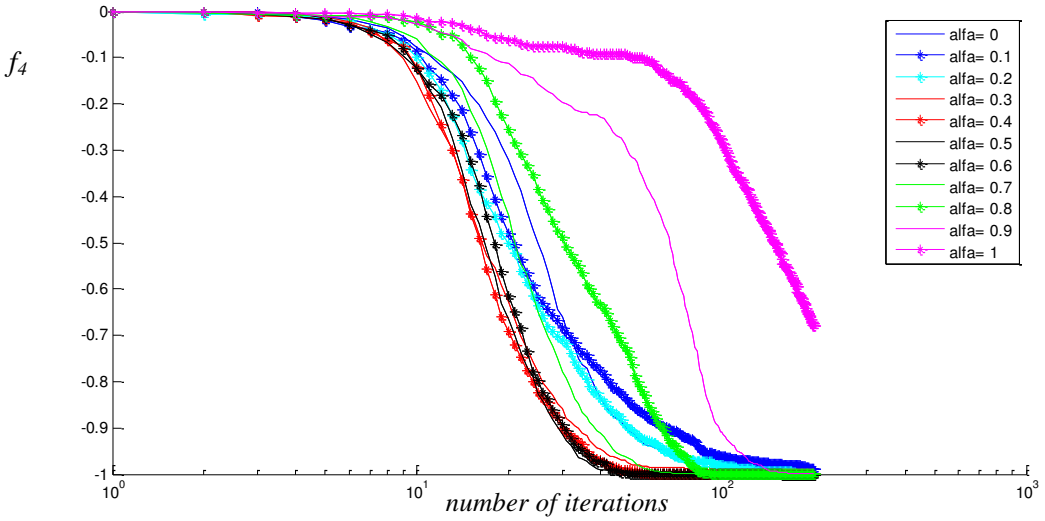

Figure 4: Evolution of the *Easom* function changing α.

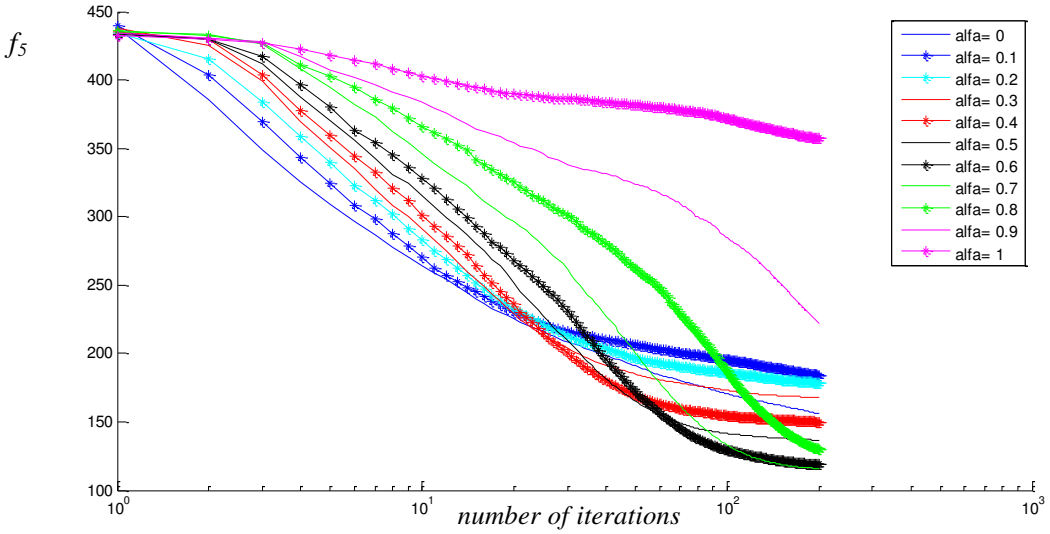

Figure 5: Evolution of the *Rastrigin* function changing α.

Experimental results show that the convergence of the algorithm depends upon the fractional order α. However, contrarily to [14] work, the Darwinian algorithm easily avoids being stuck in local solutions independently on the value of α (since it is a particularity of the traditional *DPSO*).

5 Conclusions

The search for an algorithm capable of dealing with most optimization problems without being very time-consuming and computationally demanding has been a subject of research in several scientific areas such as control engineering and applied mathematics. Fractional calculus has appeared as a tool to enhance the performance of conventional mathematical methods.

This work proposed a new optimization algorithm based on the Darwinian Particle Swarm Optimization using the concept of fractional derivative to control the convergence

rate. Experimental results show that the speed of convergence of the algorithm depends on the fractional order α. However, each optimization problem may have a different optimal α. Therefore, as future work, we propose to adapt the *FO-DPSO* with adaptive ability to tune the fractional order α.

References

[1] Floreano, D. and Mattiussi, C. (2008). "*Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*". Cambridge, MA: MIT Press.

[2] Eberhart, R. and Kennedy, J. (1995). "*A New Optimizer using Particle Swarm Theory*" in Proc. 6th Int. Symp. Micro Machine and Human Science (MHS), Oct. 1995, pp. 39–43.

[3] Tillett, T., Rao, T.M., Sahin, F. and Rao, R (2005). "*Darwinian particle swarm optimization*". Proceedings of the 2nd Indian international conference on artificial intelligence, Pune, Índia, pp. 1474–1487.

[4] Sabatier, J., Agrawal, O.P., Tenreiro Machado, J.A. (eds.) (2007). "*Advances in Fractional Calculus - Theoretical Developments and Applications in Physics and Engineering*". Springer, Berlin. ISBN:978-1-4020-6041-0.

[5] Valle, Y. del, Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J. C. and Harley, R. (2008). "*Particle swarm optimization: Basic concepts, variants and applications in power systems*". IEEE Trans. Evol. Comput., vol. 12, no. 2, pp. 171–195.

[6] Tang, J., Zhu, J. and Sun, Z. (2005). "*A novel path panning approach based on appart and particle swarm optimization*". Proceedings of the 2nd International Symposium on Neural Networks, LNCS 3498, pp. 253-258.

[7] Pires, E. J. S., Oliveira, P. B. M., Machado, J. A. T. and Cunha, J. B. (2006). "*Particle Swarm Optimization versus Genetic Algorithm in Manipulator Trajectory Planning*". 7th Portuguese Conference on Automatic Control, September 11-13.

[8] Couceiro, M. S., Mendes, R. M., Ferreira, N. M. F. and Machado, J. A. T. (2009). "*Control Optimization of a Robotic Bird*". EWOMS '09, Lisbon, Portugal, June 4-6.

[9] Couceiro, M.S., Luz, J.M.A., Figueiredo, C.M., Ferreira, N.M.F. (2011a). "*Modeling and Control of Biologically Inspired Flying Robots*". Journal of Robotica - Cambridge University Press.

[10] Alrashidi, M.R. and El-Hawary, M.E. (2006). "*A Survey of Particle Swarm Optimization Applications in Power System Operations*". Electric Power Compon Syst. v34 i12, pp. 1349-1357.

[11] Couceiro, M.S., Luz, J.M.A., Figueiredo, C.M., Ferreira, N.M.F. & Dias, G. (2010). "*Parameter Estimation for a Mathematical Model of the Golf Putting*". In VM Marques, CS Pereira & A Madureira (Eds.), Proceedings of WACI-Workshop Applications of Computational Intelligence. ISEC.IPC. Coimbra. 2 de Dezembro: pp: 1-8. ISSN 978-989-8331-10-6.

[12] Shi, Y. and Eberhart, R. (2001). "*Fuzzy adaptive particle swarm optimization*". In Proc. IEEE Congr. Evol. Comput., vol. 1, pp. 101–106.

[13] Secrest, B. and Lamont, G. (2003). "*Visualizing particle swarm optimization - Gaussian particle swarm optimization*". In Proc. IEEE Swarm Intell. Symp., pp. 198–204.

[14] Pires, E.J.S., Machado, J.A.T., Oliveira, P.B.M., Cunha, J.B., Mendes, L. (2010). "*Particle swarm optimization with fractional-order velocity*". Journal on Nonlinear Dynamics, 61: 295–301.

[15] Blackwell, T. and Bentley, P. (2002). "*Don't push me! Collision-avoiding swarms*". in Proc. IEEE Congr. Evol. Comput., vol. 2, pp. 1691–1696.

[16] Krink, T., Vesterstrom, J., and Riget, J. (2002). "*Particle swarm optimization with spatial particle extension*". in Proc. IEEE Congr. Evol. Comput., vol. 2, pp. 1474–1479.

[17] Miranda, V. and Fonseca, N. (2002). "*New evolutionary particle swarm algorithm (EPSO) applied to voltage/VAR control*". in Proc. 14th Power Syst. Comput. Conf.

[18] Lovbjerg, M. and Krink, T. (2002). "*Extending particle swarms with self-organized criticality*". in Proc. IEEE Congr. Evol. Comput., vol.2, pp. 1588–1593.

[19] Chia-Feng, J. (2004). "*A hybrid of genetic algorithm and particle swarm optimization for recurrent network design*". IEEE Trans. Syst., Man, Cybern., Part B: Cybern., vol. 34, no. 2, pp. 997–1006.

[20] Angeline, P. (1998). "*Using selection to improve particle swarm optimization*". In Proc. IEEE Int. Conf. Evol. Comput., pp. 84–89.

[21] Zhang, W. and Xie, X. (2003). "*DEPSO: Hybrid particle swarm with differential evolution operator*". In Proc. IEEE Int. Conf. Syst., Man, Cybern., vol. 4, pp. 3816–3821.

[22] Kannan, S., Slochanal, S. and Padhy, N. (2004). "*Application of particle swarm optimization technique and its variants to generation expansion problem*". ELSERVIER Electric Power Syst. Res., vol. 70, no. 3, pp. 203–210.

[23] Couceiro, M.S., Rocha, R.P., Ferreira, N.M.F. (2011b). "*A Novel Multi-Robot Exploration Approach based on Particle Swarm Optimization Algorithms*". The IEEE/RSJ International Conference on Intelligent Robots and Systems September, 25-30, 2011, San Francisco, California (Under Review).

[24] Ortigueira, M. D., & Tenreiro Machado, J. A. (2003). "*Special Issue on Fractional Signal Processing*". Signal Process, 83, 2285- 2480.

[25] Machado, J.A.T., et al. (2010). "*Some Applications of Fractional Calculus in Engineering*". Hindawi Publishing Corporation Mathematical Problems in Engineering, 2010, 1-34.

[26] Podlubny, I. (1999). "*Fractional Differential Equations*". Mathematics in Science and Engineering, 198, San Diego, California, Academic Press.

[27] Debnath, L. (2003). "*Recents Applications of Fractional Calculus to Science and Engineering*". Int. J. Math. Math. Sci., 54, 3413- 3442.

[28]Elshehawey, E. F., Elbarbary, E. M. E., Afifi, N.A.S., & El-Shahed M. (2001). "*On the Solution of the Endolymph Equation Using Fractional Calculus*". Appl. Math. Comput., 124, 337-341.

[29] Figueiredo, R., Camargo, A.O., Chiacchio, & Capelas de Oliveira, E. (2008). "*Differentiation to fractional orders and the fractional telegraph equation*". Journal of Mathematical Physics, 49, 033-505.

[30] Bergh, F.V. den, Engelbrecht, A.P. (2006). "*A Study of Particle Swarm Optimization Particle Trajectories*". Inf. Sci. 176(8), 937–971.